Computer Science and Engineering & Information Technology (2^{nd} Year B.Tech.)
IIIT Kalyani, West Bengal

Operating System Lab (CS 411): (Spring: 2019-2020)

Assignment - 7 *Marks: 10*
Assignment Out: 6^{th} *March, 2020*

1. Write a C++ program to show the *race condition* on shared data in the following way:
   (a) The program reads a positive integer n, the the dimension of two vectors.
   (b) It creates two 1-D array of size n and of type `double` in the *heap* (not on stack).
   (c) Declares a global variable `dotProd` of type `double` to compute the dot-product of the vectors.

$$\texttt{dorProd} \leftarrow \sum_{i=0}^{n-1} \vec{u}[i] \times \vec{v}[i].$$

   The variable is initialize to zero (0.0).
   (d) The program creates three threads using Linux `clone()` (not pthread).
   (e) The thread-1 computes the dot-product of the lower one-third of u and v i.e. index 0 to $n/3$, adds it to `dotProd`. Similarly the thread-2 computes the dot-product of the middle one-third and the thread-3 computes the dot-product of the remaining portion of the array. These values are added to `dotProd`. Each thread uses the **same thread function** with different parameters.
   (f) Finally the main thread prints the value of the dot-product.
   (g) As three threads are accessing the same location `dotProd` there is a possibility of race. Amplify the the possibility of race by introducing a *delay* at an appropriate place in the thread function. Run the code with the *delay* and without the *delay*.

Input:

```
$ ./a.out
Enter the dimension of the vector: 3
Enter the first vector: 1 2 3
Enter the second vector: 10 20 30
With or without delay (1/0): 0
```

Output:

```
vector-1: 1 2 3
vector-2: 10 20 30
Dot product: 140
```

Input:

```
$ ./a.out
Enter the dimension of the vector: 3
Enter the first vector: 1 2 3
Enter the second vector: 10 20 30
With or without delay (1/0): 1
```

Output:

```
vector-1: 1 2 3
vector-2: 10 20 30
Dot product: 90
```

Input:

```
$ ./a.out
Enter the dimension of the vector: 3
Enter the first vector: 1 2 3
Enter the second vector: 10 20 30
With or without delay (1/0): 1
```

Output:

```
vector-1: 1 2 3
vector-2: 10 20 30
Dot product: 10
```

2. Write a C++ program to compare the time to create a process and a thread. Use Linux `clone()` to create a process and also to create a thread. Both of them may execute the same function e.g. computation of of factorial. But we are interested only about the *creation time*, not the completion time of a thread or a process.

   You may use clock time function `int clock_gettime(clockid_t clk_id, struct timespec *tp);` to get the real-time in micro seconds before and after an event.

   You may also use inline-assembly code similar to the following one to get the value of the time-stamp counter before and after an event.

```
 __asm__ __volatile__(
              "cpuid \n\t"
              "rdtsc \n\t"
              "shl $32, %%rdx\n\t"
              "orq %%rdx, %%rax\n\t"
              :"=a" (start)
    ) ;
// the C++ program variable 'start' is of type 'long long unsigned'.
```

Run

```
$ a.out
Enter a +ve integer: 5
Thread creation time stamp count: 164534
Thread creation time: 48.77 microSec
5! = 120
Process creation time stamp count: 424154
Process creation time: 125.158 microSec
5! = 120

$ a.out
Enter a +ve integer: 5
Thread creation time stamp count: 201640
Thread creation time: 59.723 microSec
5! = 120
Process creation time stamp count: 462434
Process creation time: 142.316 microSec
5! = 120
```